also looks for the pTOC (where TOC represents table of contents) and RomExt variables for later processing. When pTOC is found, a new .creloc header is created with the bSrcSection set to the section retrieved from the map file (adjusted from 1 based to 0 based), the bDstSection set to 254 and a length of four. This header is written to the target file followed by the 4 byte RVA+Base address information. For RomExt, the header is the same except that bSrcSection is set to 253.

[**0081**] The pTOC/RomExt parsing operation is used because the operating system kernel requires information about the files in the ROMs, which is provided through the pTOC variable. This variable needs to be updated by the DiskImage tool and the update application running on the device. The information about this variable can only be retrieved through the .MAP file created during the compile and link phase of a system build. This file is parsed to retrieve this information.

[**0082**] Certain runtime tools need access to a variable declared within the kernel data structures. This variable is named RomExt. This variable needs to be updated by the DiskImage tool and the Update Application running on the device. The information about this variable can only be retrieved through the .MAP file created during the compile and link phase of a system build, and thus this file is parsed to retrieve this information.

[**0083**] In accordance with an aspect of the present invention, for a package to be self-describing, a device manifest file **260** (**FIG. 2**, shown in more detail in **FIG. 9**) is created during the packaging process and stored in the package itself. The device manifest file **260** is used during the installation process. The format and information contained in the device manifest file is shown in **FIG. 9** and is also described with this structure definition:

```
typedef struct _DeviceManifestHeader
{
      const DWORD dwStructSize;          // Size of this structure (in bytes)
                                         // for versioning
      const DWORD dwPackageVersion;      // Version of this package
      const DWORD dwPrevPkgVersion;      // Version of package that this package
                                         // updates. (0) for Canonical
      const DWORD dwPackageFlags;        // package specific identifiers.
      const DWORD dwProcessorID;         // what processor (matches defines in
                                         // winnt.h)
      const DWORD dwOSVersion;           // what version of the operating system
                                         // was this built to.
      const DWORD dwPlatformID;          // what was the target platform.
      const DWORD dwNameLength;          // length of filename in bytes.
      const DWORD dwNameOffset;          // offset to Friendly name of package
      const DWORD dwDependentCount;      // How many entries in Dependent GUID
                                         // list.
      const DWORD dwDependentOffset;     // How many bytes from the front of the
                                         // file are the dependent GUID structs.
      const DWORD dwShadowCount;         // How many entries in shadow GUID list.
      const DWORD dwShadowOffset;        // How many bytes from front of file is
                                         // the array of shadowed package GUIDs.
      const DWORD dwFileCount;           // How many files are there listed in
                                         // this manifest.
      const DWORD dwFileListOffset;      // How many bytes from the front of file
                                         // to the first FileEntry.
      const DWORD cbCERTData;            // number of bytes of digital
certificate
                                         // data
      const DWORD dwCERTDataOffset;      // How many bytes from the front of file
                                         // to the certificate data.
      const GUID guidPackage;            // GUID of this package
}DeviceManifestHeader, *PDeviceManifestHeader;
typedef struct _DependentEntry {
      const DWORD size;
      const DWORD version;
      const GUID guid;
}DependentEntry, *PDependentEntry;
typedef struct _FileEntry {
      const DWORD dwNameLength;
      const DWORD dwFlags;
      const DWORD dwOffset;
      const DWORD dwBase; // Base address that file was originally linked with
      const DWORD dwFileSize; // Size of the whole file. Not accurate for
                              // update packages.
}FILEENTRY, *PFILEENTRY;
```